Modeling Web Requests: A Multifractal Approach*

Abdullah Balamash and Marwan Krunz

Department of Electrical & Computer Engineering

University of Arizona

Tucson, AZ 85721

{balamash,krunz}@ece.arizona.edu

Abstract

WWW caching is used to improve network latency and bandwidth usage by storing previously requested files in a cache. Ideally, the cache replacement policy should account for the intrinsic characteristics of WWW traffic, which include temporal locality, spatial locality, and popularity. In this paper, we accurately capture these three characteristics in a stochastic model, which can be used to generate synthetic WWW traces and assess WWW cache designs. To capture temporal and spatial localities, we use a modified version of Riedi et al.'s multifractal model, where we reduce the complexity of the original model from $\mathcal{O}(N)$ to $\mathcal{O}(1)$; N being the length of the synthetic trace. Our model has the attractiveness of being parsimonious (characterized by few parameters) and that it avoids the need to apply a transformation to a self-similar model (as often done in previously proposed models), thus retaining the temporal locality of the fitted traffic. Furthermore, because of the scale-dependent nature of multifractal processes, the proposed model is more flexible than monofractal (self-similar) models in describing irregularities in the traffic. Trace-driven simulations are used to demonstrate the goodness of the proposed model in terms of generating representative WWW streams and approximating the cache performance of actual WWW traffic.

keywords — WWW modeling, web caching, multifractals, stack distance, self-similarity.

I. Introduction

As the World Wide Web (WWW) continues to grow at an astounding rate, its clients are experiencing excessive response times and its servers are frequently stretched to their capacity limits. One way to deal with this problem is to use WWW caching. Caches are placed either in the proximity of clients to reduce their perceived latency or close to servers to reduce their loads. To be of any practical benefit, the cache should only store the most popular documents. Determining which documents are important

* This work was supported by the National Science Foundation under grants ANI 9733143, CCR 9979310, and ANI 0095626, by a scholarship from the government of Saudi Arabia through King Abdul-aziz University, by a grant from the University of Arizona (Proposition 301 initiative), and by the Center for Low Power Electronics (CLPE) at the University of Arizona. An abridged version of this paper was presented at the *Web Engineering Workshop*, Pisa, Italy, May 2002.

and which ones are to be flushed out in case of cache saturation is the primary function of a cache replacement policy. An ideal replacement policy tries to take into account the intrinsic properties of WWW traffic in its design. These properties include temporal locality, spatial locality, and popularity. Temporal locality measures the closeness in time between requests to the same document. Spatial locality measures the correlation between requests to different documents (e.g., if document A is currently being requested, then there is a good chance that document B will be requested in the near future). Popularity refers to the overall likelihood of requesting a particular document, independent of other documents.

The ability to assess the performance of WWW caching policies hinges on the availability of a representative workload that can be used in trace-driven simulations [4], [13]. Measured ("real") traces can be used for this purpose. However, due to the difficulty associated with capturing real traces, only a handful of such traces are available in the public domain (see [19] for public domain traces). This makes it hard to provide simulation results with reasonable statistical credibility. A more feasible alternative is to rely on synthetic traces that are derived from an approximate stochastic model. The need for such a model is the main motivation behind our work.

In this paper, we present a modified version of Riedi et al.'s multifractal model [17]. We use this modified version to simultaneously capture the temporal and spatial localities of WWW traffic. Riedi's model has the attractiveness of being able to simultaneously approximate the (lognormal) marginal distribution and the correlation structure of the traffic. Its main disadvantage is its complexity, which grows linearly with the size of the generated trace. We modify this model, reducing its complexity to $\mathcal{O}(1)$. The resulting model is parsimonious in that it is characterized by four to five parameters representing the mean, variance, and correlation structure of the "normalized stack distance" string (explained below). The popularity profile of the traffic is incorporated in the model during the trace generation phase, assuming that the popularity profiles for all documents are given beforehand. Our model is mainly intended for offline generation of the traffic demand *seen by a WWW server*. Accordingly, the popularity profiles can be easily computed from the server logs.

The rest of the paper is organized as follows. Section II describes the related work. In Section III we describe the data sets used in our study. In Section IV we give a brief overview of Riedi et al.'s multifractal model and the modification we make to it to render it parsimonious. The proposed WWW traffic generation approach is given in Section V, followed by simulation studies in Section VI. We conclude the paper in Section VII

II. Related work

WWW traffic modeling has been the focus of several previous studies; examples of which are given in [1], [3], [7], [15], [14]. In these studies, the temporal locality of the traffic was represented by the marginal distribution of the stack distance string. This distribution was found to follow a lognormal-like shape. The stack distance string, which is an equivalent representation of a reference string, is obtained by transforming the reference string using the least recently used (LRU) stack , as follows. Let the reference string be $R_t = \{r_1, r_2, ..., r_t\}$, where r_j is the document (or object) requested at time j. Note that a document may appear multiple times in R_t. Let the LRU stack at time t be $S_t = \{Obj_1, Obj_2, Obj_3, ..., Obj_n\}$, where $Obj_1, Obj_2, \ldots, Obj_n$ are distinct documents; Obj_1 is the most recently requested document, Obj_2 is the second most recently requested document, and so on. Let d_t be the stack distance of the document referenced at time t (the position of the document in the LRU stack at time $t-1$). Whenever a reference is made to a document, the LRU stack must be updated. If $r_{t+1} = Obj_i$, then the LRU stack becomes $S_{t+1} = \{Obj_i, Obj_1, Obj_2, \ldots, Obj_{i-1}, Obj_{i+1}, \ldots, Obj_n\}$ and $d_{t+1} = i$. Thus, for any reference string $R_t = \{r_1, r_2, \ldots, r_t\}$, there is a corresponding stack distance string $D_t = \{d_1, d_2, \ldots, d_t\}$.

In [1] the authors showed that spatial locality can be captured (at least, in part) through the autocorrelation structure (ACF) of the stack distance string. They argued that the stack distance string exhibits long-range dependence (LRD) behavior. Thus, to simultaneously model the marginal distribution (temporal locality) and the correlation structure (spatial locality) of the stack-distance string, they relied on the work in [12], which proved the invariance of the Hurst parameter to transformations of the marginal distribution of an LRD process. More specifically, the authors in [12] proved that under some mild assumptions, a point-by-point transformation $Y = F_y^{-1}(F_x(X))$ of a *Gaussian* self-similar process X with Hurst parameter H results in a self-similar process Y with the same Hurst parameter, where F_x and F_y are the CDFs for X and Y, respectively. It should be noted, however, that this result is valid asymptotically and only for Gaussian processes (e.g., fractional ARIMA). More importantly, while this result assures the invariance of H, it does not necessarily preserve the shape of the ACF. As an example, consider the transforming of the Gaussian distribution of a F-ARIMA model into a lognormal distribution, which adequately models the marginal distribution of the stack distance string. The resulting ACFs are shown in Figure 1, along with the ACF of a real WWW stack distance string. The figure illustrates the two main drawbacks of the transformation. First, while the transformation may capture the asymptotic behavior of the ACF (the H parameter), it destroys the overall shape of the original ACF of the F-ARIMA model. Second, the original F-ARIMA model itself is not accurate in representing the real ACF at finite lags.

To avoid the problems stated above, we resort to multifractal modeling to simultaneously capture the correlation structure and the marginal distribution of the stack distance string. Figure 2 shows the accuracy of the multifractal model (which we describe in Section IV) in capturing the ACF of the stack-distance string of a WWW trace. Multifractality is a generalization of self-similarity (monofractality), whereby the Hurst parameter (the scaling exponent) is not fixed, but varies with scale. This variability makes multifractal processes more flexible than monofractal processes in describing "irregularities" in the traffic (e.g., contrasting short-term and long-term behaviors). The reader is referred to [8], [9], [18], [11], [17], [10] and the references therein for comprehensive discussions of multifractal processes. In [17] the authors used a wavelet-based construction of a multifractal

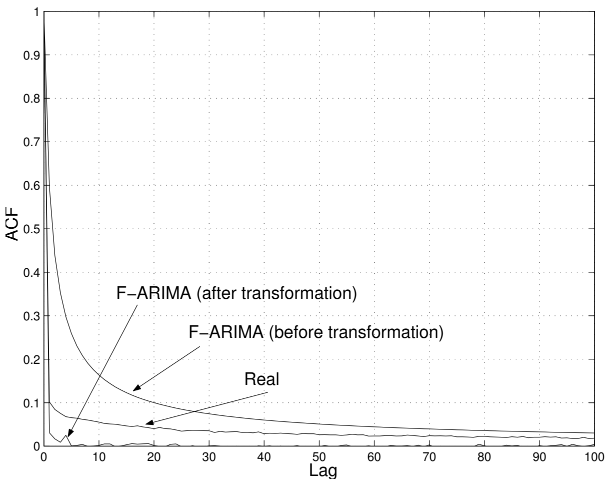

Fig. 1. Impact of transforming the marginal distribution of a F-ARIMA model on the correlation structure.

process to show that the correlation behavior of a strongly correlated time series can be approximately captured by appropriately setting the second moments of the wavelet coefficients of the multifractal process. This result provides the basis for modeling the ACF of the stack distance string. Combined with the fact that the above multifractal model exhibits an approximately lognormal marginal distribution, it can be used to model both the temporal and spatial localities in WWW traffic.

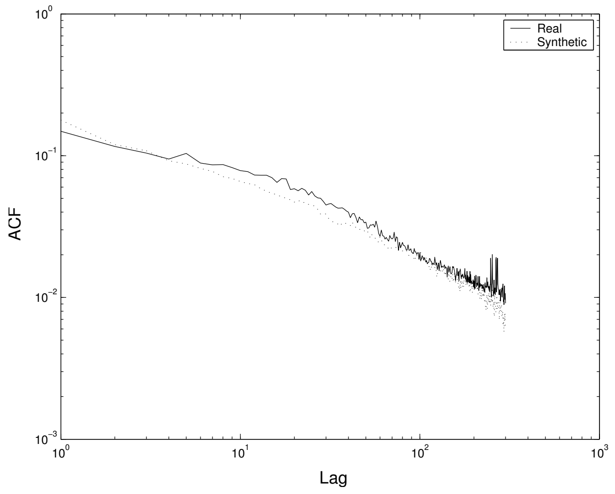

Fig. 2. Accuracy of the multifractal model in capturing the ACF of the stack distance string of a real WWW trace.

In [16], the authors studied the temporal locality in WWW traffic and concluded that such phenomenon is induced by both temporal correlations and long-term popularity. More specifically, references to long-term popular documents tend to be close to each other in time. Moreover, references to certain unpopular documents (in the long term) exhibit strong temporal correlations, whereby these references appear “clustered” in time (e.g., a document that is extensively requested but only during a short period of time). It is important to differentiate between the two aspects of temporal locality since this can help in cache design [15], [16]. The standard stack-distance-string approach does not differentiate between the two sources of temporal locality, since its distribution is predominantly affected by the popularity profile (i.e., long-term popularity). To solve this problem, there is a need to equalize the effect of popularity. The authors in [5] introduced a new measure for temporal locality, called the *scaled stack distance*, to deal with this problem. The scaled stack distance string is a normalization of the stack distances by their expected values (assuming that requests to a given document are evenly distributed over the duration of the trace). This normalization is a measure of short-term temporal correlations and is insensitive to long-term popularity, which by itself is captured during the trace generation process. For our WWW traffic model, we use the same measure, which we refer to as the *normalized stack distance*. Accordingly, equally popular documents have the same expected stack distance. The normalized stack distance string was found to have a lognormal-like distribution and a slowly decaying correlation structure (i.e., LRD behavior). We employ the multifractal model to capture both the marginal distribution and the correlation structure of the normalized stack distance string. We use extensive simulations to evaluate the performance induced by our WWW traffic model and contrast it with the self-similar model in [1] and the model in [5], using the original (real) traces as a point of reference. Our evaluation measures include sample statistics of the synthetic traces (e.g., mean, variance, correlations, percentiles) as well as the cache and byte hit ratios for a trace-driven LRU cache. The results indicate marked improvement in accuracy when using the proposed multifractal-based WWW model. We hope that this model can be subsequently used in cache design and evaluation studies.

III. Experimental Data

We briefly describe the three data sets used in the modeling study. These sets were obtained from three separate WWW servers: the Computer Science Department WWW server at the University of Calgary, the WWW server at ClarkNet (a commercial Internet provider in Baltimore, Washington DC), and from the Worldcup98 WWW servers [19]. Table I provides a summary of the main features of the data sets. More details can be found in [19], [2]. Note that the three traces have contrasting loads (in requests/second). The Calgary's load is the lightest while the Worldcup98's load is the heaviest.

	Trace		
Feature	Calgary	ClarkNet	Worldcup98
Log duration	one year	one week	One day
Start date	Oct 24, 1994	August 28, 1995	May 6, 1998
Log size (MB)	52.3	120.1	107
Total number of requests before reduction	726,739	1,164,868	1,193,353
Total number of requests after reduction	567,519	1,125,092	1,033,567
Number of unique files	8,220	20,168	3,824
Number of files referenced only once	1,752	5,279	665

TABLE I
SUMMARY OF THE DATA SETS USED IN THE MODELING STUDY.

The data sets contain several pieces of information, including the name of the host that generated the URL request, the day and time the request was recorded, the name of the requested file, the HTTP reply code (explained below), and the number of transferred bytes in response to the request. Four types of HTTP reply codes were recorded: *successful*, *not modified*, *found*, and *unsuccessful*. A *successful* code indicates that the requested file was found at the server and was returned to the client. The client may have a copy of a file, but may want to verify if this copy is up-to-date or not. If the file is up-to-date, the server responds with a *not modified* code. The *found* code indicates that the requested file is available at a different server whose address is provided in the response. Finally, the *unsuccessful* code indicates that the requested file is not available, the client has no permission to access the file, or that there is an error. In our analysis, we only included the requests with *successful* code, since they are the ones that result in actual data transfer from the server. We also excluded dynamic files (e.g., cgi and pl files).

IV. Multifractal Analysis of WWW Traffic

As indicated earlier, multifractality is a generalization of monofractality (self-similarity), where the fixed (scale independent) H parameter of a self-similar process is now scale dependent. The variability in the H value gives added flexibility to multifractal processes, allowing them to characterize irregularities in the data being modeled. Furthermore, certain multifractal processes, including the one considered in this paper, inherently exhibit an approximately lognormal-like marginal distribution, in line with the shape of the (fitting) marginal distribution of typical WWW traces. This convenient feature allows us to avoid the risky step of transforming the marginal distribution, leaving us with the task of fitting the ACF. In this section, we first briefly describe Riedi et al.'s multifractal model [17]. This model uses a wavelet-based construction to approximately capture the correlation behavior of a given time series by appropriately setting the second moments of the wavelet coefficients at each scale. Its main deficiency is its complexity, which grows linearly (in the number of parameters) with the size of the generated trace. We then describe how we modify this model to reduce its complexity to $\mathcal{O}(1)$, and then we apply the modified model to characterize the temporal and spatial localities of WWW traffic.

A. Riedi et al.'s Multifractal Model

Riedi et al.'s model relies heavily on the discrete wavelet transform. The idea behind the wavelet transform is to express a signal (time function) $X(t)$ by an approximated (smoothed) version and a detail. The approximation process is repeated at various levels (scales) by expressing the approximated signal at a given level j by a coarser approximation at level $j-1$ and a detail. At each scale, the approximation is performed through a scaling function $\phi(t)$, while the detail is obtained through a wavelet function $\psi(t)$. More formally, a wavelet expansion of the signal $X(t)$ is given by:

$$X(t) = \sum_k U_{J,k}\phi_{J,k}(t) + \sum_{j=J}^{\infty}\sum_k W_{j,k}\psi_{j,k}(t) \tag{1}$$

where

$$W_{j,k} \triangleq \int_{-\infty}^{\infty} X(t)\psi_{j,k}(t)dt \tag{2}$$

$$U_{j,k} \triangleq \int_{-\infty}^{\infty} X(t)\phi_{j,k}(t)dt \tag{3}$$

and $\psi_{j,k}$ and $\phi_{j,k}$, $j,k = 0,1,2,\ldots$, are *shifted* and *translated* versions of the wavelet and scaling functions $\psi(t)$ and $\phi(t)$, respectively, and are given by:

$$\psi_{j,k}(t) \triangleq 2^{-j/2}\psi(2^{-j}t-k) \tag{4}$$

$$\phi_{j,k}(t) \triangleq 2^{-j/2}\phi(2^{-j}t-k). \tag{5}$$

In (1), the index J indicates the coarsest scale (the lowest in detail). The coefficients $W_{j,k}$ and $U_{j,k}$ are called, respectively, the wavelet and scale coefficients at scale j and time $2^j k$. Together, they define the discrete wavelet transform of the signal $X(t)$, assuming that $\phi(t)$ and $\psi(t)$ are specified. Several wavelet and scale functions have been used in the literature, giving rise to different wavelet transforms. One popular (and simple) transform is the Haar wavelet transform. This transform, which is specified by the coefficients $W_{j,k}$ and $U_{j,k}$ for all j and k, can be obtained recursively as follows (we adopt the same convention of [17], where the higher the value of j, the better is the approximation of the original signal):

$$U_{j,k} = \frac{U_{j+1,2k} + U_{j+1,2k+1}}{\sqrt{2}} \tag{6}$$

$$W_{j,k} = \frac{U_{j+1,2k} - U_{j+1,2k+1}}{\sqrt{2}} \tag{7}$$

To initialize the recursion, the values of $U_{j,k}$, $k = 0,1,\ldots,2^j-1$, at the highest value of j are taken as the empirical trace to be modeled. Figure 3 depicts the generation process of the scale coefficients (from top to bottom).

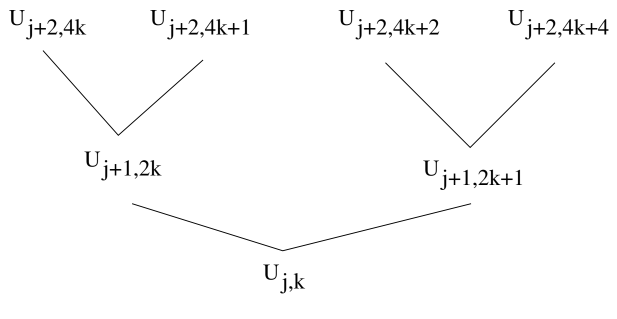

Fig. 3. Process of generating the scaling coefficients in the DWT.

In order to generate synthetic traces with a given autocorrelation structure, the Haar transform is reversed by rewriting (6) and (7) as:

$$U_{j+1,2k} = \frac{U_{j,k} + W_{j,k}}{\sqrt{2}} \tag{8}$$

$$U_{j+1,2k+1} = \frac{U_{j,k} - W_{j,k}}{\sqrt{2}} \tag{9}$$

Now to generate nonnegative data, which in our case represent the stack distance string, we need to have $|W_{j,k}| \leq U_{j,k}$. To satisfy this constraint, the wavelet coefficients can be defined as:

$$W_{j,k} = A_{j,k} U_{j,k} \tag{10}$$

where $A_{j,k}$ is a random variable (rv) defined on the interval $(-1, 1)$. Using (8), (9), and (10), the following recursion can be obtained for synthesizing the scale coefficients:

$$U_{j+1,2k} = (\frac{1 + A_{j,k}}{\sqrt{2}})U_{j,k} \tag{11}$$

$$U_{j+1,2k+1} = (\frac{1 - A_{j,k}}{\sqrt{2}})U_{j,k} \tag{12}$$

The rvs $A_{j,k}$ must also satisfy the following additional constraints [17]:

1) $A_{j,k}, k = 0, 1, ..., 2^j - 1$, are *i.i.d* rvs that can be represented by the generic rv A_j having the same CDF as $A_{j,k}$.
2) For each j, the probability density function of the rvs $A_{j,k}, k = 0, 1, \ldots, 2^j - 1$, is symmetric with zero mean.
3) A_j is independent of A_l for $l > j$ and is also independent of $U_{0,0}$.

The wavelet energy at a given scale is given by the variance of the wavelet coefficients at that scale. It has been shown that the correlation structure of the signal can be approximately captured by controlling the wavelet energy decay across scales [17]. The ratio of the energy at scale $j-1$ to the one at scale j (j is finer than $j-1$) was found to be [17]:

$$\eta_j = \frac{E[W_{j-1}^2]}{E[W_j^2]} = 2\frac{E[A_{j-1}^2]}{E[A_j^2](1 + E[A_{j-1}^2])} \tag{13}$$

Assuming that $E[W_j^2]$ is given for all j, Equation (13) can be used to solve for $E[A_j^2]$, $j = 1, 2, \ldots$. The recursion can be initialized using $E[A_0^2] = \frac{E[W_0^2]}{E[U_0^2]}$, where W_0 and U_0 are the wavelet and scale coefficients at the coarsest scale.

In [17], the authors suggested two different distributions for A_j. One of them is a symmetric beta distribution that has the following pdf:

$$f_{A_j}(x) = \frac{(1+x)^{\rho_j - 1}(1-x)^{\rho_j - 1}}{\beta(\rho_j, \rho_j) 2^{2\rho_j - 1}} \tag{14}$$

where ρ_j is the parameter of the rv and $\beta(.,.)$ is the beta function. The variance of this random variable is given by:

$$\text{var}[A_j] = \frac{1}{2\rho_j + 1}. \tag{15}$$

The other distribution is a point-mass distribution defined as:

$$\Pr[A_j = c_j] = \Pr[A_j = -c_j] = r_j$$

$$\Pr[A_j = 0] = 1 - 2r_j$$

In the case of a beta distributed A_j, the parameter ρ_j at each scale can be found by solving (13) and (15), resulting in:

$$\rho_j = \frac{\eta_j}{2}(\rho_{j-1} + 1) - 1/2 \tag{16}$$

This, however, assumes that $E[W_j^2]$ is given for $j = 1, 2, 3, \ldots$. Since η_j, $j = 1, 2, \ldots$, cannot be obtained using a parametric model, it must be computed from the empirical data, which makes the number of fitted parameters in the model in the order of N; N being the trace length.

On the other hand, if A_j has a point-mass distribution, then (13) by itself is not sufficient to compute both parameters of A_j (c_j and r_j). An alternative approach for computing these parameters is to rely on the following expression for the moments of the scaling coefficients at different scales:

$$\frac{E[U_j^q]}{E[U_{j-1}^q]} = 2^{-q/2} E[(1 + A_{j-1})^q], \quad q = 1, 2, \ldots \tag{17}$$

However, to apply (17) one needs to have two moments (i.e., two values for q) for each scale j. Again, unless we can compute these values using a parametric model, we have to rely on the empirical data to do so, which makes the model more complex than if a beta distributed A_j were to be used.

With either distribution of A_j, it was shown in [17] that the above model generates positive-valued autocorrelated data with an approximately lognormal marginal distribution.

B. Reducing the Number of Parameters

As shown in the previous section, whether A_j is a beta or a point-mass rv, one needs to provide the second moments of the wavelet coefficients or two moments of the scale coefficients at each scale in order to completely determine A_j, $j = 1, 2, \ldots$. This significantly increases the complexity of the model, as the number of parameters to be computed a priori is in the order of the trace length. Moreover, the point-mass rv is not rich enough and takes only three possible values.

To reduce the complexity of the model, we select A_j to be a continuous-valued rv with one parameter. Then using (17) with $q = 2$, we compute the parameter of A_j, $j = 1, 2, \ldots$, assuming that we can compute the ratio $E[U_j^2]/E[U_{j-1}^2]$ using a small number of parameters (the mean μ, the variance σ, and the correlation structure of the modeled data), as shown later. The selection of the rv A_j will be discussed in Section IV-C.

For a discrete time series $X = \{X_i : i = 1, 2, \ldots\}$, we define $X^{(m)} = \{X_i^{(m)} : i = 1, 2, \ldots\}$ to be the aggregated time series of X at aggregation level m:

$$X_n^{(m)} = \sum_{i=nm-m+1}^{nm} X_i, n = 1, 2, 3, \ldots, N/m \tag{18}$$

where $m = 1, 2, 4, 8, ...N$; N is the length of X. Note that if the aggregation level m corresponds to scale j, then the aggregation level $2m$ corresponds to scale $j - 1$. From the definition of the Haar wavelet transform, the following holds:

$$\frac{E[(X^{(m)})^q]}{E[(X^{(2m)})^q]} = 2^{-q/2} \frac{E[U_j^q]}{E[U_{j-1}^q]}, \quad \text{for } q = 1, 2, \ldots \tag{19}$$

From (19) and (17) we get:

$$\frac{E[(X^{(m)})^q]}{E[(X^{(2m)})^q]} = 2^{-q} E[(1 + A^{(2m)})^q] \tag{20}$$

where $A^{(2m)} = A_{j-1}$. Evaluating (20) at $q = 2$, we obtain the following expression:

$$E[(A^{(2m)})^2] = 4 \frac{E[(X^{(m)})^2]}{E[(X^{(2m)})^2]} - 1 \tag{21}$$

To reduce the number of parameters in the multifractal model, we need to analytically obtain $E[(X^{(m)})^2]$ for all possible values of m. The variance at aggregation level m, var$[X^{(m)}] \triangleq V^{(m)}$, can be expressed in terms of the autocorrelation function of the signal [6]:

$$V^{(m)} = mv + 2v\sum_{k=1}^{m}(m-k)\rho_k \tag{22}$$

The mean, $E[(X^{(m)})] = \mu^{(m)}$, is given by:

$$\mu^{(m)} = m\mu \tag{23}$$

where μ and v are the mean and variance of the original signal, respectively. The second moment of $X^{(m)}$ is then given by:

$$E[(X^{(m)})^2] = mv + 2v\sum_{k=1}^{m}(m-k)\rho_k + m^2\mu^2 \tag{24}$$

From (21) and (24), the parameter of the rv A_j can be computed for all scales $j = 1, 2, \ldots$, given μ, v, and the correlation structure of the time series being modeled. For normalized stack distance strings, we found that the form $\rho_k = e^{-\beta\sqrt[n]{g(k)}}$, $k = 0, 1, \ldots$, fits the correlation structure very well, where g is a function of the lag k. For both the ClarkNet and the Worldcup98 traces, $g(k) = k$ produced a good fit to the empirical ACF, while for the Calgary trace, $g(k) = \log(k+1)$ was found appropriate. Figure 4 shows the fitting of the ACF functions for the three traces.

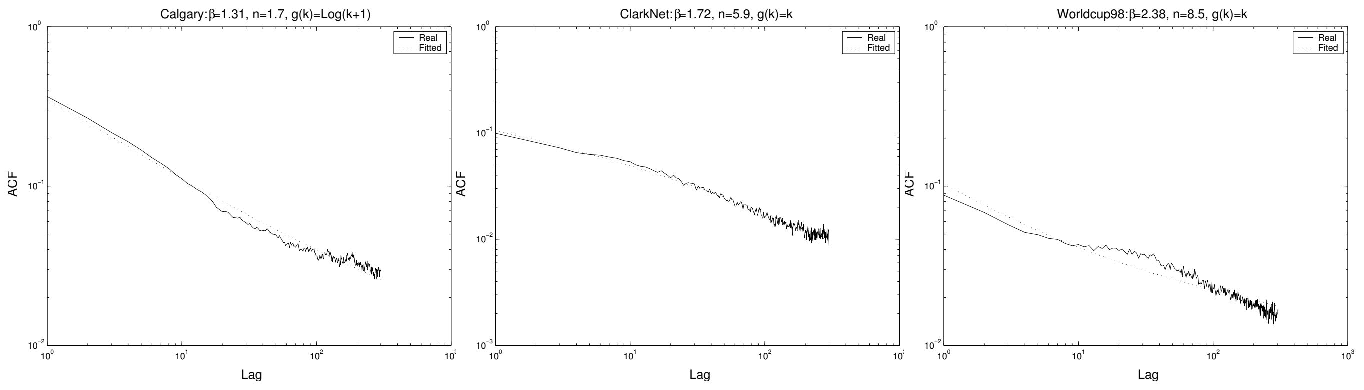

Fig. 4. Fitting the correlation structure of the normalized stack distance string.

In summary, to use the multifractal model for modeling the normalized stack distance string, we only need five parameters:

- Mean of the normalized stack distance string (μ).
- Variance of the normalized stack distance string (v).
- Autocorrelation structure (parameterized by β, n, and g).

Using these parameters, along with (24) and (21), one can compute the parameter of the rv A_j at each aggregation level (scale).

The synthesis process starts from the highest level of aggregation. At this level we can start with l data points that are normally distributed with mean $m_h\mu$ (the mean at aggregation level m_h) and variance of var$[X^{(m_h)}]$, where m_h is the highest aggregation level. After that, the process can be carried out using (11) and (12).

C. Selecting the Random Variable A_j

As indicated earlier, the rv variable A_j must be symmetric with zero mean and defined on the interval (-1,1). To reduce the number of parameters of the multifractal model, we require that the A_j is specified by one parameter only. There are many rvs that can satisfy these conditions. The difference between one rv and another is the range of the values that the ratio $E[(X^{(m)})^2]/E[(X^{(2m)})^2]$ can take. It can be shown that for any distribution, this ratio satisfies the following inequalities:

$$0.25 \leq \frac{E[(X^{(m)})^2]}{E[(X^{(2m)})^2]} \leq 0.5 \tag{25}$$

The lower bound follows immediately from (21) and the fact that $E[(A^{(2m)})^2] \geq 0$. The proof for the upper bound is provided in the appendix. As an example, consider the uniform rv in the range $[-c, c]$, where $|c| < 1$. The variance of this rv is given by $V = c^2/3$. Solving (21) for c, we get:

$$c = 3(4\frac{E[(X^{(m)})^2]}{E[(X^{(2m)})^2]} - 1).$$

Since $c < 1$,

$$\frac{E[(X^{(m)})^2]}{E[(X^{(2m)})^2]} < \frac{1}{3}$$

Using similar calculations, Table II shows the upper bound for a number of popular rvs.

Random variable	Upper bound
Symmetric beta(ρ, ρ)	0.5000
Uniform$(-c, c)$	0.3333
Triangular$(-c, 0, c)$	0.2917
Normal$(0, \sigma)$	0.2778

TABLE II
UPPER BOUND ON THE RATIO $E[(X^{(m)})^2]/E[(X^{(2m)})^2]$ FOR VARIOUS DISTRIBUTIONS.

For the three traces we use in this work, the ratio $E[(X^{(m)})^2]/E[(X^{(2m)})^2]$ did not exceed 0.3333, and as a result, we decided to use the uniform rv since it is the simplest one.

V. MODELING WWW TRAFFIC

In this section, we describe our approach for modeling the stream of file objects generated by a WWW server. Let U be the number of unique files (or objects) at the server and let fr_i be the fraction of times that the ith file, $i = 1, 2, ..., U$, appears in the reference string (fr_i is the popularity profile of file i). The modeling approach proceeds in three steps. First, we extract the stack distance string from the URL reference string. Then, we apply some form of normalization to capture both sources of temporal locality (temporal correlation and long-term popularity). The modified multifractal model described in the previous section is then applied to model the normalized stack distance string. Finally, we incorporate the popularity profile of the traffic during the process of generating synthetic reference strings. These steps are described next.

A. *Extracting the Empirical Normalized Stack Distance String*

In our model, we use the concept of stack distance to model the temporal and spatial localities in WWW traffic. The authors in [3] extract the stack distances from the original trace assuming an arbitrary initial ordering of the stack. Whenever an object is requested, its depth in the stack (stack distance) is recorded and the object is pushed to the top of the stack. In our model, we avoid making any assumptions on the initial ordering of the stack, which we have found to disturb the marginal distribution and the correlation structure of the stack distance string. We start with an empty stack and process the empirical reference string *in the reverse direction*, starting from the last reference. If a file is referenced for the first time (in the reverse direction), it is put on top of the stack but no stack distance is recorded. Otherwise, if the file has already been referenced before (hence, it is already in the stack), then it is pushed from its previous location in the stack to the top of the stack and its depth is recorded as a stack distance. Finally, the resulting trace of stack distances is reversed to get the correct stack distance string. The following example illustrates the idea. Consider the reference string [a d c b c d d a b], where each letter indicates the name of a file. If we process this string starting from the end, the first reference is to file b. Since this is the first time file b is being referenced, we push it to the top of the stack without recording any distance. The same procedure is performed for the next two references (for files a and d). The fourth reference (from the end) is for file d. Since this file has been referenced before, it gets pushed to the top of the stack and its stack depth is recorded (in this case, the stack depth for file d is one). The procedure continues until all references are processed (see Figure 5). The end result of this process is the stack distance stream [4 3 2 4 1].

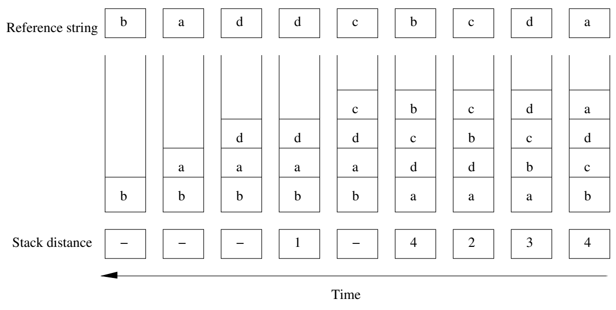

Fig. 5. Example showing our approach for extracting stack distances from a real trace.

Temporal locality in a stream of WWW requests is attributed to two factors: long-term popularity and short-term temporal correlations. Both factors are important for cache design [15], [14], and must therefore be incorporated in the model for temporal locality. In [15], it was found that the (lognormal) distribution of the stack distance string is predominantly affected by the popularity profile (i.e., long-term popularity). So the marginal distribution of the stack distance (without normalization) does not capture the effect of short-term temporal correlations.

To accurately capture the temporal locality of the traffic, we need to isolate the effect of popularity from that of short-term correlations. One solution is to have a separate stack-distance-string model for equally popular objects. Another solution, used in our work, is to capture how much a stack distance deviates from its "expected" value. This deviation is captured by normalizing (scaling) the stack distances by their expected values. The resulting scaled stack distance string is a measure of how objects are clustered over the trace length (temporal correlation). This measure is insensitive to the popularity profile, so it allows us to separately model the popularity profile and the short-term temporal correlations.

The expected stack distance for a file i is computed as follows:

$$E[d_i] = \sum_{g \neq i} \frac{f_g}{f_g + f_i - 1} \tag{26}$$

where f_i is the number of references to file i [5]. For the three studied traces, we found that the normalized stack distance string has an approximately lognormal marginal distribution. Figure 6 shows the fitting of the lognormal marginal distribution of the normalized stack distance string.

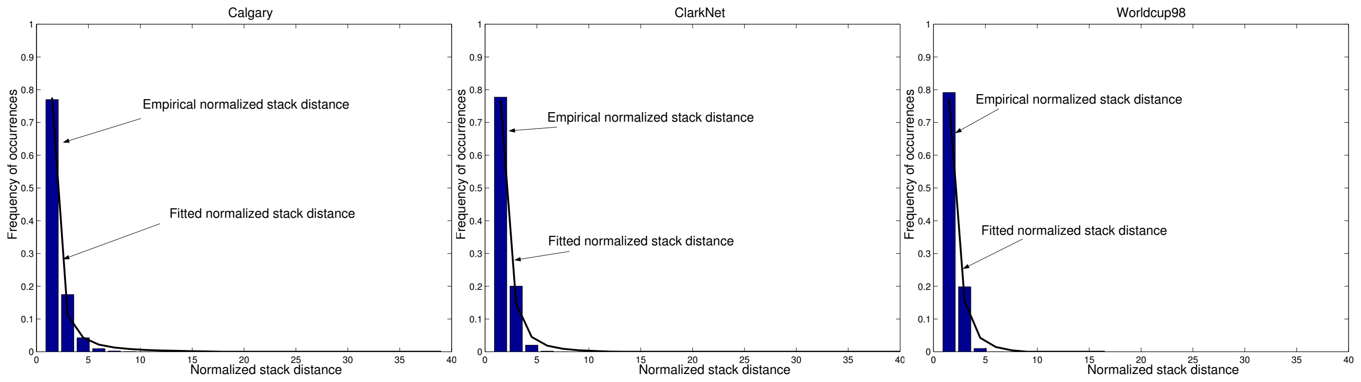

Fig. 6. Marginal distribution of the normalized stack distance string.

B. *Modeling the Normalized Stack Distance String*

To model the normalized stack distance string, we need to determine μ, v, β, and n. Once the values of these parameters are determined, the multifractal model described in Section IV can be used to capture the marginal distribution and the correlation structure of the normalized stack distance string. Note that spatial locality is captured by modeling the autocorrelation function of the *unscaled* stack-distance string. However, since the multifractal model is applied to the scaled version of the stack-distance string, we have to invert back the synthesized trace of the multifractal model so that the resulting trace models the unscaled version of the data. The pseudo-code in Figure 7 describes the modeling process. The function GenNormStackDistance takes five input parameters. The first three parameters are μ, v, and the ACF of the normalized stack distance string to be generated. The fourth parameter, N, is the length of the synthetic normalized stack distance string. The fifth parameter, l, is the number of data points at the coarsest scale. GenNormStackDistance starts by computing the number of aggregation levels, $NumAggLevels$ (line 2). In line 4, the second moment of the normalized stack distance string at aggregation level $m = 1$ is computed. The for loop starting in line 5 is used to compute the second moment for the remaining aggregation levels. The for loop that starts from line 8 computes the summation in (24). In line 11, the second moments at higher aggregation levels are computed using (24). The second moments ratio, $E[(X^{(m)})^2]/E[(X^{(2m)})^2]$, is computed in line 12. The parameter of the rv $A^{(2m)}$ is computed in line 13 using (21) and substituting for the variance of the uniform rv in $E[(X^{(2m)})^2]$. After computing these parameters, the mean of the time series at the highest aggregation level is computed in line 15 using (23). The variance at the highest aggregation level is computed in line 16. In line 17, l data points are generated to represent the coarsest level. The while loop starting from line 19 is used to continue the generation process using (11) and (12).

```
**GenNormStackDistance**($\mu, v, \rho_k, N, l$)
2   $NumAggLevels = \lceil \log_2(N/l) \rceil$
3   $m = 1$
4   $SecMom(1) = mv + m^2\mu^2$
5   **for** $i = 2$ to $NumAggLevels$ **Do**
6       $m = 2m$
7       $sum = 0$
8       **for** $k = 1$ to $m$ **Do**
9           $sum = sum + (m - k)\rho_k$
10      **end for**
11      $SecMom(i) = mv + 2vs + m^2\mu^2$
12      $SecMom\_ratios(NumAggLevels - i + 1) = \frac{SecMom(i-1)}{SecMom(i)}$
13      $AParm(NumAggLevels - i + 1) = \sqrt{3(4\ SecMom\_ratio(NumAggLevels - i + 1) - 1))}$
14  **end for**
15  $\mu_h = 2^{(NumAggLevels-1)} * \mu$
16  $V_h = SecMom(NumAggLevels) - \mu_h^2$
17  $NormStkDist = norm\_random(\mu_h, \sqrt{V_h}, l)$
18  $i = 1$
19  **While**(length($NormStkDist$) $< N$) **Do**
20      $A = Uniform\_random(-AParm(i), AParm(i), length(NormStkDist))$
21      $NormStkDist = [NormStkDist\ \frac{(1+A)}{2}\ NormStkDist\ \frac{(1-A)}{2}]$
22      $i = i + 1$
23  **end while**
**END**
```

Fig. 7. Algorithm for generating synthetic scaled stack distance strings.

C. Modeling Popularity and Generating Synthetic Reference Strings

To generate a synthetic WWW reference string, we first generate a synthetic normalized stack distance string, as shown in the previous section. The process of generating a synthetic WWW reference string starts by arranging the unique documents of the WWW server in an LRU stack. This is done by sampling from a probability distribution that is weighted by the popularity profiles of the various documents (i.e., the more popular a document is, the more likely it will be placed closer to the top of the stack). This ordering approach was used in [5]. It is known to provide more accurate results than using an arbitrary ordering. Note that even though the probability of selecting a *given* unpopular document is small, the probability of selecting *any* of the unpopular documents is relatively large (because of the large number of unpopular documents). So, probabilistically, there is a good chance that *some* unpopular documents will be placed near the top of the stack. To generate a reference string of length N, we first compute the number of references a document can get according to its popularity profile. Then, the top document in the LRU stack is considered as the next referenced document in the synthetic reference string. If the required number of references for this document is reached, then this document is flushed out of the stack. Otherwise, it is pushed down the stack according to the next value in the normalized stack distance string. This is done after scaling back the normalized stack distance by multiplying it by the corresponding expected stack distance for the object in hand (objects with the same popularity profile have the same expected stack distance). Note that our notion of a "stack" allows for the insertion of an object in between two objects in the stack, which does not happen in a regular LRU stack. This process continues until the popularity profiles of all objects are satisfied (no documents are left in the LRU stack). The pseudo-code in Figure 8 describes the generation process. Function GenTrace accepts three parameters: the synthetic normalized stack distance ($NormStkDist$), the number of requests each file gets (req), and the lru stack with the files ordered according to the popularity profile (i.e., placed randomly according to the empirical distribution of the popularity profiles). The while loop in line 4 is used to generate the reference string. Line 5 is used to record the next reference, $Ref(i)$, taken as the file at the top of the LRU stack. Then the number of outstanding references to this file, $req(Ref(i))$, is reduced by one (line 7). If this number reaches zero, then the file is dropped out the stack. Otherwise, the next stack distance, $StkDist$, is computed in line 11 by scaling back the normalized stack distance according to the popularity of the file. The file is then pushed $StkDist$ positions down the stack. The while loop in line 4 is continued until the LRU stack is empty.

VI. Experimental Results

In this section, we evaluate the accuracy of the proposed multifractal model and contrast it with two other models. The first model is a self-similar (monofractal) model [1], [3]. This model involves transforming the Gaussian marginal distribution of a fractional ARIMA process into a lognormal distribution. We simply refer to this model as the LRD model. The second model

```
GenTrace($NormStkDist, req, lru$)
2   $i = 0$
3   $k = 0$
4   While(stack is not empty) Do
5      $Ref(i)$=top($lru$)
6      $i = i + 1$
7      $req(Ref(i)) = req(Ref(i)) - 1$
8      if ($req(Ref(i)) == 0$) then
9         drop document from the stack
10     else
11        $StkDist$ = Scale-Back($NormStkDist(k)$,$Ref(i)$)
12        Push-Top-Document-Down-Stack($StkDist$)
13        $k = k + 1$
14     endif
15  end while
END
```

Fig. 8. Algorithm for generating synthetic WWW strings.

was proposed by Cherkasova et al. [5]. The three investigated models were mainly designed for offline traffic generation, with the primary purpose of generating synthetic traces for use in cache design studies. Accordingly, we compare these models in terms of the file and byte miss ratios seen at an LRU cache that is driven by synthetic traces from these models. The comparison is made with reference to the cache performance seen under the real traffic. The results are shown in Figures 9, 10, and 11.

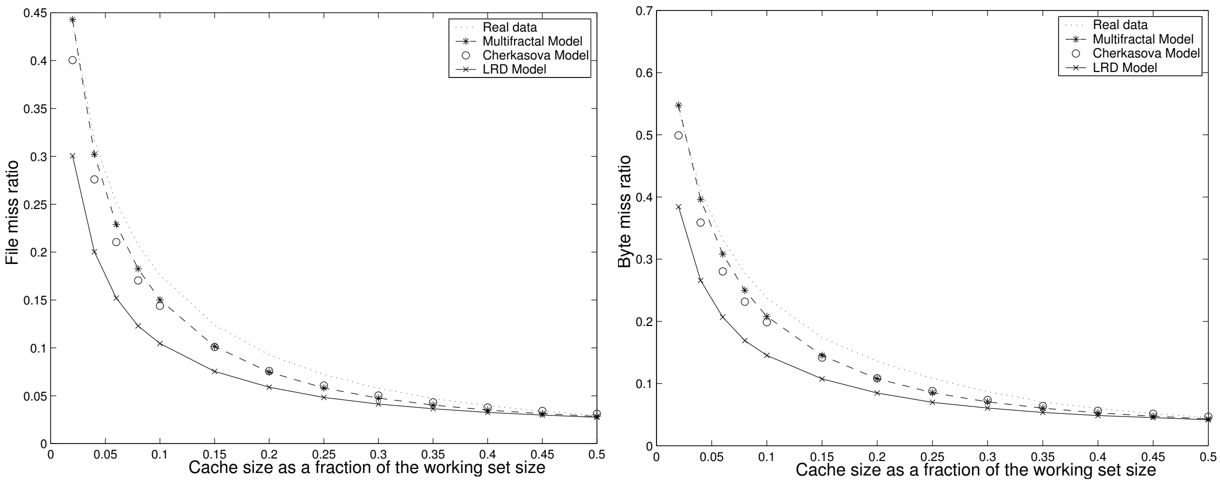

Fig. 9. File/byte miss ratio versus cache size (Clarknet trace).

It is clear that of the three models, the proposed multifractal model produces the most accurate results, especially for small cache sizes. The relative accuracy in terms of capturing the behavior of the real data is greater in the case of the Calgary data. Consider, for example, the Calgary data with a normalized cache size of 0.3. The percentage inaccuracies in the file miss rate for the multifractal model, the LRD model, and Cherkasova et al.'s model are 0.5%, 53%, and 111%, respectively. In the case of the byte miss rate, the corresponding values are 4.9%, 65%, and 109%. The overall improvement in the accuracy of the file and byte miss rates due to the use of the multifractal model is significant. Moreover, our model captures the spatial locality

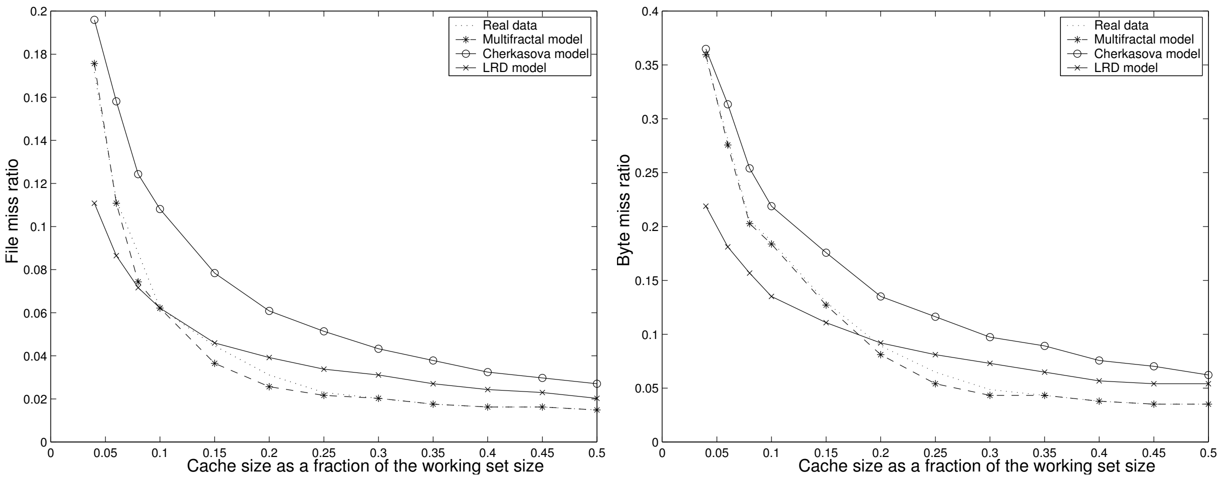

Fig. 10. File/byte miss ratio versus cache size (Calgary trace).

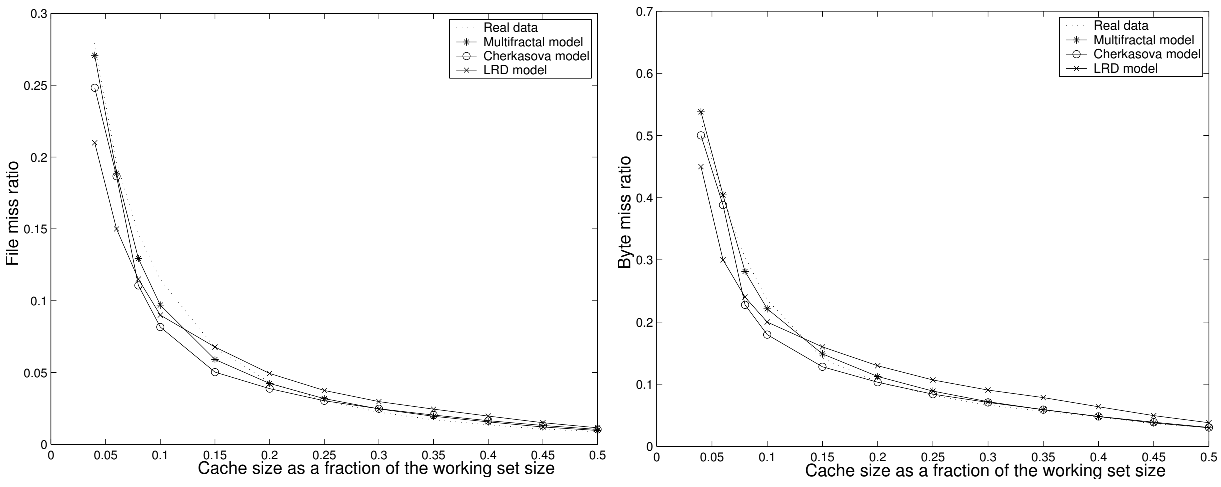

Fig. 11. File/byte miss ratio versus cache size (Worldcup98 trace).

which is not reflected through the file/byte miss ratios of the LRU caching policy. To show how well our model captures this property, we also compared the inter-request times (number of requested files between any two references to a given file) in the real data to those in the synthetic traces of the models. We found that the sequence of inter-request times is non-stationary (has a clear decreasing trend). So we normalized this sequence by the *expected* inter-request time, which is equal to $1/fr_i$, for file i. The mean, variance, percentile values, and some values of the autocorrelation function for the normalized inter-request times are shown in Tables III,IV, and V. Note that the mean, variance, percentile values of the inter-request times are measures of the temporal locality, while the autocorrelation is a measure of the spatial locality.

From these results, it is clear that the multifractal model is more accurate than the other two models in capturing these statistics.

Statistics		Real data	Multifractal model	Cherkasova's model	LRD model
μ		0.954	0.937	0.806	0.905
σ		1.032	1.164	1.918	1.428
ρ_1		0.130	0.118	0.000	0.060
ρ_5		0.076	0.075	0.000	0.020
ρ_{10}		0.061	0.058	0.000	0.009
ρ_{25}		0.039	0.039	0.000	0.001
Percentiles	75%	1.306	1.149	0.778	1.063
	90%	2.233	2.210	1.697	2.161
	98%	3.976	4.230	4.980	5.051

TABLE III
STATISTICAL COMPARISONS FOR CLARKNET TRACE.

Statistics		Real data	Multifractal model	Cherkasova's model	LRD model
μ		0.714	0.68	0.62	0.907
σ		1.541	2.010	7.080	4.224
ρ_1		0.192	0.185	0.000	0.030
ρ_5		0.070	0.063	0.000	0.030
ρ_{10}		0.036	0.036	0.000	0.027
ρ_{25}		0.009	0.010	0.000	0.023
Percentiles	75%	0.813	0.813	0.771	0.326
	90%	1.790	1.790	2.293	2.046
	98%	4.289	5.241	5.531	6.644

TABLE IV
STATISTICAL COMPARISONS FOR CALGARY TRACE.

Statistics		Real data	Multifractal model	Cherkasova's model	LRD model
μ		0.990	0.951	0.823	0.941
σ		0.986	1.137	1.718	1.289
ρ_1		0.054	0.054	0.000	0.098
ρ_5		0.027	0.037	0.000	0.089
ρ_{10}		0.025	0.030	0.000	0.071
ρ_{25}		0.022	0.023	0.000	0.034
Percentiles	75%	1.373	1.136	0.856	1.134
	90%	2.266	2.099	1.727	2.101
	98%	3.843	4.210	4.666	4.726

TABLE V
STATISTICAL COMPARISONS FOR WORLDCUP98 TRACE.

VII. Conclusions

In this work, we demonstrated the potential of multifractal processes as a viable approach for WWW traffic modeling. We started with the multifractal model of Riedi et al., which is capable of generating approximately lognormal synthetic traces with any desired autocorrelation structure. However, to apply this model in traffic fitting and trace generation, one needs to match as many parameters of the model as the length of the trace to be generated. To make the model parsimonious, we modified it by using a different distribution for the multiplier A_j (which relates the wavelet and scale coefficients) and by analytically expressing the parameter of $A_j,\ j = 1, 2, \ldots$, in terms of the mean, variance, and ACF of the modeled data. As a result, the modified multifractal model is specified by five parameters only. We fitted this model to the normalized stack distance strings of three WWW traffic traces. The proposed model captures the spatial and temporal localities of the real traffic as well as the popularity profile. Trace-driven simulations of the LRU cache policy indicates that the proposed model gives much more accurate cache miss rates than two previously proposed WWW traffic models. Statistics of the normalized inter-request distances support the goodness of our model. Our future research will focus on designing new cache replacement and prefetching policies that exploit the characteristics of the traffic and that rely on model predictions in making file replacement and prefetching decisions.

Appendix

Let $X \overset{\triangle}{=} \{X_i : i = 1, 2, \ldots\}$ be a positive valued stationary random process, and let $Y \overset{\triangle}{=} \{Y_i : i = 1, 2, \ldots\}$ be an aggregation of X that is defined as follows:

$$Y_n = X_{2n-1} + X_{2n}$$

Note that $X^{(m)}$ in (25) represents X, while $X^{(2m)}$ represents Y. We now prove that $E[X_n^2]/E[Y_n^2] \leq 0.5$.

$$\begin{aligned}
\frac{E[X_n^2]}{E[Y_n^2]} &= \frac{E[X_n^2]}{E[(X_{2n-1} + X_{2n})^2]} \\
&= \frac{E[X_n^2]}{E[X_{2n-1}^2 + 2X_{2n-1}X_{2n} + X_{2n}^2]} \\
&= \frac{E[X_n^2]}{E[X_{2n-1}^2] + 2E[X_{2n-1}X_{2n}] + E[X_{2n}^2]}
\end{aligned}$$

Since X is stationary, then $E[X_{2n-1}^2] = E[X_{2n}^2] = E[X_n^2]$, which leads to:

$$\begin{aligned}\frac{E[X_n^2]}{E[Y_n^2]} &= \frac{E[X_n^2]}{2E[X_n^2]+2E[X_{2n-1}X_{2n}]} \\ &= \frac{1}{2+2\frac{E[X_{2n-1}X_{2n}]}{E[X_n^2]}} \leq 0.5\end{aligned}$$

since $E[X_{2n-1}X_{2n}]/E[X_n^2] \geq 0$.

References

[1] V. Almeida, A. Bestavros, M. Crovella, and A. Oliverira. Characterizing reference locality in the WWW. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS)*, pages 92–103, 1996.

[2] M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of the ACM SIGMETRICS Conference*, pages 126–137, 1996.

[3] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the ACM SIGMETRICS Conference*, pages 151–160, 1998.

[4] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 USENIX Symposium on Internet Technology and System*, pages 193–206, 1997.

[5] L. Cherkasova and G. Ciardo. Characterizating temporal locality and its impact on web server performance. In *Proceedings of the Ninth International Conference on Computer Communication and Networks (ICCCN)*, pages 434–441, 2000.

[6] D. Cox. Long-range dependence: A review. *Statistics: An Appraisal*, pages 55–74, 1984. The Iowa State University, Ames, Iowa.

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW client-based traces. *IEEE/ACM Transactions on Networking*, 1(3):134–233, Jan 1999.

[8] A. Feldman, A. Gilbert, W. Willinger, and T. Kurtz. The changing nature of network traffic: Scaling phenomena. In *Proceedings of the ACM SIGCOMM Conference*, number 2, pages 5–29, April 1998.

[9] J. Gao and I. Rubin. Multifractal analysis and modeling of long-range dependent traffic. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 1, pages 382–386, 1999.

[10] A. Gilbert, W. Willinger, and A. Feldmann. Scaling analysis of conservative cascades, with applications to network traffic. *IEEE Transactions on Information Theory - Special Issues of on Multiscale Statistical Signal analysis and its Applications*, 45(3):971–991, 1999.

[11] A. Gillbert and W. Willinger. Scaling analysis of conservative cascades with applications to network traffic. In *Proceedings of the ACM SIGCOMM Conference*, number 4, pages 42–55, 1998.

[12] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. R. Kays. Modeling and simulation of self-similar variable bit rate compressed video: A unified approach. In *Proceedings of the ACM SIGCOMM Conference*, volume 25, pages 114–125, 1995.

[13] S. Jin and A. Bestavros. Popularity-aware greedy-dual size web proxy caching algorithms. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, Taiwan, May 2000.

[14] S. Jin and A. Bestavros. Sources and characteristics of web temporal locality. In *Proceedings of IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Fransisco, CA, August 2000.

[15] S. Jin and A. Bestavros. Temporal locality in web request streams. In *Proceedings of the ACM SIGMETRICS Conference*, pages 110–111, 2000.

[16] S. Jin and A. Bestavros. Greedy-dual* web caching algorithm. *International Journal on Computer Communications*, 24(2):174–183, Febreuary 2001.

[17] R. Riedi, M. Crouse, V. Ribeiro, and R. Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE Transactions on Information Theory*, 45(3):992–1018, April 1999.

[18] R. Riedi. Introduction to multifractals. http://www.dsp.rice.edu/publications/.

[19] Internet traffic archive at http://ita.ee.lbl.gov/.